積項の長さに制限を付けた論理関数のOrdered Tree-Shellability (計算機科学基礎理論の新展開)

AUTHOR(S):

東海林, 貴司; 武永, 康彦

CITATION:

東海林, 貴司 ...[et al]. 積項の長さに制限を付けた論理関数のOrdered Tree-Shellability (計算機科学基礎理論の新展開). 数理解析研究所講究録 2004, 1375: 130-136

ISSUE DATE:

2004-05

URL:

http://hdl.handle.net/2433/25586

数理解析研究所講究録 1375巻 2004年 130-136

積項の長さに制限を付けた論理関数の Ordered Tree-Shellability

電気通信大学大学院情報工学専攻 東海林 貴司 (Takashi Toukairin) *

電気通信大学情報工学科 武永 康彦 (Yasuhiko Takenaga) †

1 はじめに

論理関数の特性を理解し，明らかにすることは，計算機科学の様々な分野において重要なことである．積和形論理式 (Disjunctive Normal Form Boolean Formula: DNF) は，論理関数の表現方法の1つである．それぞれの積項は論理関数の内項 (implicant) に相当し，特に主項 (prime implicant) は，論理関数の最小形を表すなど，非常に重要な概念である．全ての項が主項である非冗長 DNF や，正のリテラルのみで構成されている積和形正論理式 (Positive DNF : PDNF) など，様々な DNF が定義されている．

Tree-shellable 論理関数 [1] とは，正論理関数において関数の主項とその関数を表す二分決定木の1ノードへたどるパスの数が等しい論理関数である．論理関数 f が tree-shellable であれば，f の双対関数 f^d を求めるのは容易であり，f の DNF 表現から f^d の非冗長な DNF 表現を求めることは多項式時間で可能である．このように関数が tree-shellable 論理関数であれば計算時間を短縮することができる．Ordered tree-shellable 論理関数は tree-shellable 論理関数の一種であり，関数を表す二分決定木が順序付き二分決定木となっているものである．

本研究では与えられた関数が，ordered tree-shellable 論理関数であるかどうかの判定の複雑さについて考える．しかし，ordered tree-shellable 関数であるか否かの判定は一般には NP 完全である．そこで，関数に制約を加えることで，多項式時間で判定可能な関数のクラスについても研究が行なわれている．例えば，積和形の論理関数において，各項が2つのリテラルのみで構成されている関数 (quadratic 関数) は，tree-shellable 論理関数，ordered tree-shellable 論理関数の判定はともに，多項式時間で可能であることが証明されている．本研究では，quadratic 関数に1つ，又は2つ積項を加えた関数に対しても，多項式時間で判定可能であることを証明する．

本稿は，次のように構成されている．第2章では，基本的な表現，積和形論理式や二分決定木の定義を行なう．第3章では様々な tree-shellable 論理関数の定義や特性を述べる．第4章では，quadratic 関数に対しての判定複雑さを紹介し，第5章では，quadratic 関数を拡張した論理関数に関しての判定複雑さの証明を行う．最後に第6章では考察，および今後に解決すべき課題を挙げる．

2 準備

2.1 基本表記

n が自然数であるとし，$[n] = \{1, 2, \cdots, n\}$ であるとする．ただし，$[0] = \emptyset$ である．π を整数の全順序とし，$\pi(i)$ を π の i 番目の要素とする．π に関して，s が t の前に現れることを $s \prec_\pi t$ と表す．$S \subseteq [n]$ において，順序 π について $h \in S$ かつ，任意の $i \in S \setminus \{h\}$ について $h \prec_\pi i$ であるとき，$\min_\pi(S) = h$，$h \in S$ かつ，任意の $i \in S \setminus \{h\}$ について $i \prec_\pi h$ であるとき，$\max_\pi(S) = h$ と定義する．π が明らかなときには，単に $s \prec t$，$\min(S)$，$\max(S)$ と書く．また，$S, T \subseteq [n], S \cap T = \emptyset$ に対し、$S \prec_\pi T$ という表記は S の任意の要素が T の任意の要素よりも前にくる，という意味である．

*Department of Computer Science and Infomation Mathematics,Graduate School of Electro-Communications,the University of Electro-Communications.

†Department of Computer Science,the University of Electro-Communications.

2.2 二分決定木

二分決定木は論理関数を表す，ラベル付けされた木である．二分木の葉ノードには0か1，他のノードには変数がラベル付けされている．葉ノード以外のノードからは2本の枝が出ており，それぞれ0枝，1枝と呼ばれる．ノード v から出た0枝，1枝が同じノードを指しているとき，v は冗長ノードといい，冗長ノードを持たない二分木を reduced 二分木という．変数の全順序 π が存在し，二分木の全てのパスにおいて，現れる変数の順序が π に矛盾なく従っている場合，その二分木を順序付き二分木という．π を順序付き二分木の変数順序と呼ぶ．変数ノード v が $x_{\pi(i)}$ とラベル付けされているとき，i を v のレベルと呼び，$level(v)$ と表す．根ノードから v までのノードの数を v の深さと呼び，$depth(v)$ と表す．全ての変数ノード v について，$level(v) = depth(v)$ が成り立つとき，その順序付き二分木は *leveled* であるという．二分木の全てのノードの数を，二分木のサイズと言う．

関数の値は，関数を表していてる二分決定木の根ノードから変数の値に従って枝を進み，葉ノードの定数までたどることにより得られる．葉ノードの値が1(0)であれば，その関数の値は1(0)である．以下の図では簡単のため値が0である葉ノードを省略している．また，必ず左枝を0枝，右枝を1枝とする．根ノードから値が1である葉ノードまでのパスを1パスという．根ノードから0枝のみを通っているパスを *street* という．二分決定木のパス P は変数の列で表されているとする．もし1パス P の k 番目の枝が x_i でラベル付けされたノードの1枝(0枝)を通っていたら，P の k 番目の要素は $x_i(\bar{x}_i)$ である．$P^{(k)}$ を，パス P の k 番目の要素とする．

2.3 積和形論理式

$f(x_1, \cdots, x_n)$ を論理関数とし，$I, J \subseteq [n]$ である積項 $\bigwedge_{i\in I} x_i \bigwedge_{j\in J} \bar{x}_j$ が1となるような任意の $x \in \{0,1\}^n$ について $f(x) = 1$ となるとき，$f \geq \bigwedge_{i\in I} x_i \bigwedge_{j\in J} \bar{x}_j$ と表し，積項 $\bigwedge_{i\in I} x_i \bigwedge_{j\in J} \bar{x}_j$ を，f の内項と呼ぶ．また，任意の $s \in I$ について，$\bigwedge_{i\in I-\{s\}} x_i \bigwedge_{j\in J} \bar{x}_j \not\leq f$ を満たし，任意の $t \in J$ について，$\bigwedge_{i\in I} x_i \bigwedge_{j\in J-\{t\}} \bar{x}_j \not\leq f$ を満たすとき，その内項を f の主項と呼ぶ．$f = \bigvee_{k=1}^{m} (\bigwedge_{i\in I_k} x_i \bigwedge_{j\in J_k} \bar{x}_j)$ の形で表された式を積和形論理式と呼ぶ．ただし，任意の $k(1 \leq k \leq m)$ について，$I_k, J_k \subseteq [n]$，$I_k \cap J_k = \emptyset$ とする．また，任意の k について $J_k = \emptyset$ である DNF を積和形正論理式 (PDNF) といい，f が PDNF で表されているとき，f は正論理関数という．また，DNF からある積項を取り除いてももとの関数を表している場合，その DNF を冗長な DNF といい，どの積項を除いた場合ももとの関数を表さない場合，その DNF を非冗長な DNF という．以後，DNF は非冗長な DNF であるとする．

3 Tree-Shellable 論理関数

3.1 Tree-Shellable 論理関数

定義 1 [1] f は *PDNF* $f = \bigvee_{k=1}^{m} \bigwedge_{i\in I_k} x_i$ で表されているとする．f を表す二分木で m 個の *1* パス $P_1, \cdots, P_m$ を持つものが存在するとき，f は *tree-shellable* であるという．

定理 2 [1] $f = \bigvee_{k=1}^{m} \bigwedge_{i\in I_k} x_i$ が *tree-shellable* であるとき，以下を満たす二分木 T を構成できる．

1. T は $P_1, \cdots, P_m$ の m 個の *1* パスを持つ．
2. 各々の $P_k(1 \leq k \leq m)$ は $i \in I_k \rightarrowtail x_i \in P_k$ を満たす項 I_k に対応する．

定理 3 [1] どの *1* パス P_k のどの要素 $\bar{x}_s \in P_k$ に対しても，ある t について，$P_k^{(t-1)} = P_l^{(t-1)}$，$P_k^t = \bar{x}_s$，$P_l^t = x_s$，$I_k \cup \{s\} \supset I_l$ を満たす l が存在する．

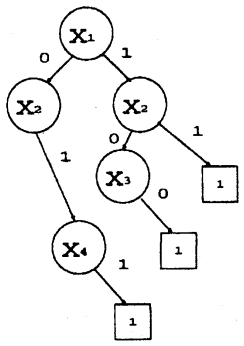

図 1: Tree-Shellable 論理関数の例 $f = x_1x_2 + x_1x_3 + x_2x_4$

3.2 Ordered Tree-Shellable 論理関数

定義 4 [1] f は $PDNF$ $f = \bigvee_{k=1}^{m} \bigwedge_{i \in I_k} x_i$ で表されているとする．f を表す変数順序が π である順序付き二分木 T で m 個の *1* パス $P_1, \cdots, P_m$ を持つものが存在するとき，f は π に関して *ordered tree-shellable* であるという．また，このような π が存在するとき f は *ordered tree-shellable* であるという．

定理 5 [1] 論理関数 $f = \bigvee_{k=1}^{m} \bigwedge_{i \in I_k} x_i$ が，任意の I_k，$i \notin I_k$, $i \prec_\pi \max(I_k)$ について *1* か *2* のいずれかが成り立つとき，f は π に関して *ordered tree-shellable* である．

1. $(I_k \cup \{i\}) \cap \{x_{\pi(1)}, \cdots, i\} = I_l \cap \{x_{\pi(1)}, \cdots, i\}$ を満たす I_l は存在しない．
2. そのような I_l が存在するならば，少なくとも *1* つは $I_k \cup \{i\} \supset I_l$ を満たす I_l が存在する．

また，論理関数 f が *(ordered)tree-shellable* である証拠となる二分決定木，とは論理関数の積項の数と等しい数の *1* パスを持つ f を表す木である．

4 Quadratic 関数に対する判定の複雑さ

ここでは，関数に制約を加えることで，多項式時間で判定可能であるクラスの関数について考える．具体的には，各項のリテラル数を制限する．各積項が 2 つのリテラルからなる関数，つまり，どのような $k \in [m]$ に対しても $|I_k| = 2$ を満たす積和形論理式 $\bigvee_{k=1}^{m} \bigwedge_{i \in I_k} x_i$ で表される関数を *quadratic* 関数という．

Quadratic 関数は，グラフ $G = (V, E)$ として表すことができる．ノードは変数，枝は項にそれぞれ相当する．$V = [n]$，項 x_i が存在するならば，$(i, j) \in E$ とする．V' により導かれる G の部分グラフ (誘導部分グラフ) $G' = (V', E')$ は次のように定義される．

$$V' \subset V,\ E' = \{(i, j) \mid i, j \in V'\}$$

誘導部分グラフが長さが $|V'|$ のサイクルを作るとき，これを誘導サイクルと言う．

長さが 4 以上の誘導サイクルを含まないグラフを triangulated グラフと言う．Triangulated グラフの補グラフを cotriangulated グラフと言い，全ての誘導部分グラフが cosimplicial ノードを含む．Cosimplicial ノードとは，そのノードから隣接していないノードによる誘導部分グラフが，独立点集合となるノードである．

定理 6 [2] *Quadratic* 関数 $f = \bigvee_{(i,j) \in E} x_i x_j$ は，$G = ([n], E)$ が *cotriangulated* グラフであるときかつそのときに限り *ordered tree-shellable* である．

変数順序は次のようにして求めることができる．全てのノードが選ばれるまで次のことを繰り返す．

1. cosimplicial ノードまたは，枝が出ていない独立したノードを選ぶ
2. 選んだノードと，そのノードから出る枝を削除する

この定理を用いて quadratic 関数 f が ordered tree-shellable である証拠となる二分木を作ることができる．

f を表すグラフを G とする．二分木の根として，G の cosimplicial ノード r_1 を一つ選ぶ．r_1 が $a, b, \cdots, l$ と辺を持っていたとしたら，r_1 の 1 枝の先に a を置く．a の 1 枝は葉ノード 1 を指し，0 枝は b を指す．b の 1 枝は葉ノード 1 を指し，0 枝は c を指す，というように l まで続ける．次に r_1 の 0 枝の先は，G から r_1 を除いた誘導部分グラフの cosimplicial ノード r_2 にする．r_2 の 1 枝，0 枝の先は r_1 と同様にする．

この操作を G の辺が無くなるまで続けることによって f を表し，tree-shellable である証拠となる二分木 T ができる．T の 1 枝側のノードの順序を入れ換えても 1 パスの本数が変わることはないので，T の street に変数の表れる順に並べ替えれば，quadratic 関数 f が ordered tree-shellable である証拠となる二分木になる．

系 7 *Quadratic* 関数 f に対して，f が *ordered tree-shellable* であるか否かの判定，および，変数順序を求めることは多項式時間で可能である．また，*tree-shellable* であるか否かの判定も多項式時間で可能である．

補題 8 *Quadratic* 関数 f が *ordered tree-shellable* であるとする．証拠となる木において，*street* の全てのノードの *1* 枝側に，ある変数 x_c が現れているならば，x_c は f を表すグラフの *cosimplicial* ノードである．

証明　f を表すグラフを G とし，x_c は G の cosimplicial ノードでないと仮定する．すなわち，c と隣接しない頂点 (v_y, v_z とする) 間に辺がある．1 枝側に x_c が現れる必要十分条件は street のノードと x_c 間に辺があることである．定理 6 のように，x_c と辺を持つ cosimplicial ノードを全て削除した後にも v_y, v_z 間の辺は残っている．このいずれかは street に現れる必要がある．したがって，street の変数の 1 枝側に x_c が現れない二分木しかできないことになり矛盾である．すなわち x_c は G の cosimplicial ノードである．　□

5 Quadratic関数を拡張した論理関数の Ordered Tree-Shellability

5.1 Quadratic関数に積項を1つ加えた場合

ここでは，quadratic 関数に変数の数が3つ以上の積項が1つだけ論理和として加わっている論理関数 F を考える．F において quadratic 関数の部分を f_q と表し，変数の数が3つ以上の積項を K とする．すなわち，$F = f_q + K$ と表せる．与えられる論理関数は非冗長であると仮定し，K に含まれる変数どうしからなる項はない．

定理 9 [5] f_q が *tree-shellable* でないなら，F は *tree-shellable* ではない．

略証 F が tree-shellable になるとすると，そこから K に対応する1パスを除いたものが f_q が tree-shellable である証拠となる二分決定木である．

いま，K は m を3以上の整数とし，$x_{k_1}, x_{k_2}, \cdots, x_{k_m}$ という変数の積とする．$x_{k_1}, x_{k_2}, \cdots, x_{k_m}$ を，K に含まれる変数と呼ぶ．f_q を表すグラフを G とする．G において，K という頂点をあらたに追加する．K は，$x_{k_1}, x_{k_2}, \cdots, x_{k_m}$ が隣接する頂点全てと辺を持つものとする．このグラフを f_q+K を表すグラフと言い，G_K とする．

定理 10 F が *ordered tree-shellable* である必要十分条件は，グラフ G_K が *cotriangulated* グラフであり，かつグラフ G_K において K が *cosimplicial* ノードであることである．

証明 (←) K が cosimplicial ノードであるならば，少なくともう一つ，K と隣接した cosimplicial ノードが存在する [3]．G は G_K からノード K を取り除いてできるグラフであるので，G_K における cosimplicial ノードは，G においても cosimplicial ノードである．K は G_K において cosimplicial ノードなので，K と隣合う cosimplicial ノードをどれだけ取り除いていったとしても K が cosimplicial ノードでなくなることはない．したがって，K と隣接する cosimplicial ノードのみを street として f_q が ordered tree-shellable である証拠となる二分木を作ることができる．この木を T_{f_q} とする．T_{f_q} の street の最後の変数の 0 枝側に積項 K に対応する1パスを1本加える．street の全ての変数ノードの1枝側に K に含まれる変数が現れていることから，定理3より，この木は F が tree-shellable である証拠となる木である．

Street の変数ノードの1枝側の変数は自由に並べかえができるので，street の変数の順序に並べかえてできるこの木は F が ordered tree-shellable である証拠となる木であると言える． □

(→) 以下の補題 11, 及び補題 12 より示される．

補題 11 F が *ordered tree-shellable* である証拠となる木において，*street* の全ての変数ノードを通り，その後一度も *0* 枝を通らずに葉ノードへ行く *1* パスが積項 K に対応しているならば，G_K において K は *cosimplicial* ノードである．

証明 F が ordered tree-shellable である証拠となる木を T_F とする．定理5より，K に対応している1パスより上の street のノードの1枝側には K に含まれる変数が少なくとも一つ存在している．ここで street をそのままにして，G_K で表される quadratic 関数を表す二分木を作ると，street に現れる全てのノードと K に含まれる変数のうち少なくとも1つが隣接している．つまり，street の全てのノードの1枝側にノード K が現れている．T_F は F が ordered tree-shellable である証拠となる木であるので，G_K は cotriangulated グラフである．そして，補題8より，G_K において K は cosimplicial ノードである □

補題 12 F が *ordered tree-shellable* である証拠となる木において，積項 K に対応する *1* パスが補題 *11* 以外の場合，G_K において K は *cosimplicial* ノードである．

証明 T_F を F が ordered tree-shellable である証拠となる木とする．K に含まれる変数を根とする T_F の部分木を T_{sub} とし，その部分木を表す式を F_{sub} とする．T_{sub} の根を x_{k_1} とする．F_{sub} は quadratic 関数と K の論理和であるから，$F_{sub} = f_q' + K$ と表す．f_q' を表すグラフを G_{sub}，$f_q' + K$ を表すグラフを G_{subK} とする．$f_q' + K$ は ordered tree-shellable であるので，f_q' も ordered tree-shellable である．T_{sub} において x_{k_1} が根であるということは，x_{k_1} が f_q' の cosimplicial ノードであったことになる．G_{sub} には x_{k_1} と辺を持たない頂点間に辺はない．すなわち存在するのは x_{k_1} と辺を持つ頂点 (距離1の頂点とする) か距離1の頂点と隣接する頂点 (距離2の頂点とする) のみである．$x_{k_2}, x_{k_3} \cdots x_{k_m}$ と x_{k_1} は辺を持たないので，$x_{k_2}, x_{k_3} \cdots x_{k_m}$ は必然的に x_{k_1} からの距離2の頂点ということになる．そして，x_{k_1} が cosimplicial ノードであるので，x_{k_1} からの距離2の頂点からは，x_{k_1} からの距離1の頂点との辺以外存在しない．G_{subK} を考えた場合，K は K に含まれる変数と隣接する頂点と辺を持つ．ということは，x_{k_1} からの距離1の頂点や，距離2の頂点は，K からの距離1の頂点，K からの距離2の頂点，となる．すなわち，K と隣接しない頂点間では辺が存在しない．つまり，G_{subK} において K は cosimplicial ノードであることがわかる．(←) の証

明により，streetの最後の変数の0枝側に積項Kに対応する1パスが現れるような木に作りかえることができる．この木は，T_{sub}と同じ関数を表す．すると，補題11より，KはG_Kにおいてcosimplicialノードである．□

よって多項式時間で判定可能である．計算時間は，quadratic 論理関数の場合とほとんど変わらず，$O(mn^2)$である．

5.2 Quadratic関数に積項を2つ加えた場合

ここでは，quadratic関数に変数の数が3つの積項が2つ論理和として加わっている論理関数Fを考える．Fにおいてquadratic関数の部分をf_qと表し，変数の数が3つの積項をKとLとする．すなわち，$F = f_q + K + L$と表せる．Kは$x_{k_1}x_{k_2}x_{k_3}$，Lは$x_{l_1}x_{l_2}x_{l_3}$とする．Fがordered tree-shellableであるかどうかの判定について考える．

補題13 *Cotriangulated*グラフにおいて「取り除いてはいけないノード」をいくつか設定し，それ以外の*cosimplicial*ノードとそのノードが持つ辺を取り除いてグラフを小さくしていくとき，最小のグラフは*cosimplicial*ノードを選ぶ順番に左右されず，一意である．

証明 ある順番で取り除けるcosimplicialノードを除いていくとグラフが最小になるとする．CotriangulatedグラフGにおけるcosimplicialノードは，Gのあらゆる誘導部分グラフにおいてcosimplicialノードである．したがって，一度，取り除いてはいけないノード以外のノードがcosimplicialノードになると，以後のグラフではいつでも取り除ける．よって，どのような順番でcosimplicialノードを取り除いても常に最小のグラフが残る．□

以下に，$F = f_q + K + L$がordered tree-shellableであるか多項式時間で判定するアルゴリズムを示す．

アルゴリズム OTS-q+2

1. f_qのグラフにおけるcosimplicialノードのうちKとLに含まれる変数両方と隣接するノードを選び，streetの次のレベルの変数とする．そのようなノードがなくなるまで繰り返す．この段階でグラフから辺がなくなった場合，$K + L$がordered tree-shellableであるか判定するだけでよい．
2. Cosimplicialノードであり，かつK(あるいはL)に含まれる変数であり，$L(K)$に含まれる変数と隣接するノードをstreetにもってくる．
3. ここではx_{k_1}を選んだとし，x_{k_1}に隣接するノードを$x_{a_1}, x_{a_2}, \cdots, x_{a_i}$とすると，ここで$F$がordered tree-shellableであるための変数順序に，
   $\{x_a, x_b, \cdots, x_i\} \prec \{x_{k_2}, x_{k_3}\}$
   という制約ができる．これを制約Aとする．
   もし，x_{k_1}がKとLに共通して含まれるcosimplicialノードである場合は7へ．
4. 制約Aを満たし，Lに含まれる変数と隣接するcosimplicialノードをstreetの次のレベルの変数とする．そのようなノードがなくなるまで繰り返し，グラフから辺が無くなったら，Fはordered tree-shellableである．グラフから辺が無くならなければ5へ．
5. グラフのcosimplicialノードの中にLに含まれる変数があるか調べる，そして，そのすべてのノードについてstreetの次のレベルの変数とした場合，変数順序制約Aを満たすかどうかを調べる．条件を満たすノードが一つもなければ2へもどり，まだ調べていないノードを選ぶ．条件を満たすノードがあった場合は6へ．
6. 5で条件を満たしたノードをstreetの次のレベルのノードとする．その変数を取り除いたグラフにもまだ辺がある場合，新たな変数順序の制約Bが生じる．7へ．
7. x_{k_1}がKとLに共通して含まれるcosimplicialノードである場合は，$K + L$がordered tree-shellableかどうかを判定すればよい．K, Lに含まれるx_{k_1}以外の変数をKは$x_{k_2}x_{k_3}$，Lは$x_{l_2}x_{l_3}$とすると，
   $\{x_a, x_b, \cdots, x_i\} \prec \{x_{k_2}, x_{k_3}, x_{l_2}, x_{l_3}\}$
   という変数順序の制約が生じる．8へ．
8. 7で生じた変数順序制約を満たしつつx_{k_1}の0枝の先のquadratic関数がordered tree-shellableかどうか判定すればよい．この部分がordered tree-shellableなら関数はordered tree-shellableである．もしこの部分がordered tree-shellableにならなければ，5へ戻る．
9. 8までで小さくなったグラフを表すquadratic論理関数が変数順序制約AとBがあるなかでordered tree-shellableかどうか判定すればよい．この部分がordered tree-shellableなら関数はordered tree-shellableである．その結果，この部分がordered tree-shellableにならなければ，5へ戻る．

5.2.1 アルゴリズムOTS-q+2の正当性の証明と計算時間の考察

アルゴリズムOTS-q+2で，関数がordered tree-shellableである，といえばordered tree-shellableなのは明らかなので，以下ではordered tree-shellableである関数は，アルゴリズムOTS-q+2を通せば，必ずそうであると判定できることを示す．

1. について
もし，アルゴリズム OTS-q+2 に反して，cosimplicial ノード以外のノードを street にした場合は明らかに F が ordered tree-shellable である証拠となる木はできない．K にも L にも隣接しない cosimplicial ノードを部分木の根ノードにした場合は，定理 5 の条件を満たさないので ordered tree-shellable である証拠となる木はできない．もし，K と L に含まれる変数両方と隣接する cosimplicial ノードがまだ存在するのに，cosimplicial ノードであり，かつ K に含まれる変数であり，L に含まれる変数と隣接するノードを street にした場合，もしくは，K と L に共通して含まれる cosimplicial ノードを street にした場合，F が ordered tree-shellable である証拠となる木 (T_F とする) を作れるならば，T_F を，アルゴリズム OTS-q+2 通りに作成した木 (T'_F とする) と同じ木に変形することが可能であることを示す．

T_F は F が ordered tree-shellable である証拠となる木である．よって T_F の street に現れる変数 x_r は f_q を表すグラフ G の，その時点で残っているグラフにおいて cosimplicial ノードである．したがって，G の，その時点で残っているグラフにおいて x_r に隣接する cosimplicial ノードが存在する．

仮定より，x_r に隣接する cosimplicial ノードの中には K と L に含まれる変数両方と隣接する cosimplicial ノードがあるはずである．もしそのようなノードがないなら，アルゴリズム OTS-q+2 でも x_r を street のノードとして選ぶはずである．したがってこのような手順でアルゴリズム OTS-q+2 に沿った木に変形していくことができる．

x_r が，cosimplicial ノードであり，かつ $K(L)$ に含まれる変数であり，$L(K)$ に含まれる変数と隣接するノードである場合，x_{k_1} に隣接するノードを $x_{a_1}, x_{a_2}, \cdots, x_{a_t}$ とすると，T_F における street の変数 x_r の 1 枝側において，
$\{x_{a_1}, x_{a_2}, \cdots, x_{a_t}\} \prec \{K(L)_2, K(L)_3\}$
という制約 C が生じている．

アルゴリズム OTS-q+2OTS-q+2 において T'_F を作る際，x_r に隣接する cosimplicial ノードを street のノードにしていく過程において，この変数順序制約に矛盾は生じない．なぜなら，制約 C において後に来なければノードは x_r に隣接しないからである．

アルゴリズム OTS-q+2 において x_r を street のノードとして選んだ後は T_F の変数順序で木を作れば変数順序が前後する 1 パスができない．よってこの変形において矛盾はおきない．x_r というノードが，K と L に共通して含まれる cosimplicial ノードだとしても，以上の議論でアルゴリズム OTS-q+2 の正当性が示される．

2. について．
この条件を満たさないノードを street にしても ordered tree-shellable である証拠となる木ができないのは明らかである．

4 . について．
制約 A を満たさないノードを street の次のレベルにしても ordered tree-shellable である証拠となる木ができないのは明らかである．また，L に含まれる変数と隣接しないノードを選んだ場合も同様である．もし，L に含まれる変数と隣接し制約 A を満たす cosimplicial ノードが存在するにもかかわらず，L に含まれる変数を street の次のレベルに選んだ木 (T_F とする) が F が ordered tree-shellable である証拠となる木場合，1. と同様の議論でアルゴリズム OTS-q+2 に沿った木に変形していくことができる．

7 . について
$K + L$ が ordered tree-shellable かどうかを判定すればよい．K, L から共通の変数が除かれた形になっているため $K + L$ は quadratic 関数で，かつ積項の数が 2 つの関数となる．もし $K + L$ が tree-shellable なら任意の変数順序で $K + L$ は ordered tree-shellable である．変数順序の制約は
x_r に隣接する変数 $\prec K, L$ に含まれる変数
の 1 通りしか生じない．

9 . について．
6 までで作られた木の street の 0 枝の先には K や L に対応する 1 パスは出現しない．すなわち，その 0 枝の先は quadratic 論理関数を表す二分木になる．つまり，変数順序に制約がある quadratic 論理関数が ordered tree-shellable かどうかを判定するということになる．その判定は補題 13 より，ここでは「取り除いてはいけないノード」を変数順序制限 A および B を満たさないノードと考えることができ，取り除いてはいけないノード以外のノードが cosimplicial ノードになると，以後のグラフではいつでも取り除くことができる．一通りの取り除く順序で調べれば F が ordered tree-shellable かどうか判定できる．すなわち多項式時間で判定可能である．

補題 13 より，1,4 を終えて残るグラフは一意である．また，7 に関しても補題 13 より，どのような順番で cosimplicial ノードを取り除いても残るグラフが一意であることから，多項式時間で判定が可能である．

5.3 Quadratic 関数に積項を複数加えた場合

全ての積項の変数の数が3以下であるような関数がordered tree-shellableかどうかを判定するために，積項の数でなく，形に制限を加えた場合を考える．

$F=f_q+K_1+K_2+K_3+\cdots+K_l$ で，$K_1,K_2,\cdots,K_l$ はそれぞれ変数の数が3つであり，f_q と共通の変数 x_{k_a} と x_{k_b} を含み，残るひとつの変数は f_q にあらわれないものとする．ここでは，$K_1=x_{k_a}x_{k_b}x_1, K_2=x_{k_a}x_{k_b}x_2,\cdots,K_l=x_{k_a}x_{k_b}x_l$ とする．

定理 14 *F が ordered tree-shellable である必要十分条件は $f_q+x_{k_a}x_{k_b}$ が ordered tree-shellable であることである．*

証明 (←) x_{k_a} または x_{k_b} と項を成す変数を $x_{m_1},x_{m_2},\cdots x_{m_i}$ (i は1以上の整数) とする．$F'=f_q+x_{k_a}x_{k_b}$ とする．F' を表す順序付二分木 $T_{F'}$ を変数順序が，

・$x_{m_1},x_{m_2},\cdots x_{m_i} \prec x_{k_a},x_{k_b} \prec$ その他の変数．

となるようにつくることができる[4]．$T_{F'}$ の根から，streetの0枝を通り，x_{k_a} かつ x_{k_b} の1枝を通っているパスの先にラベル x_1 のノードをつけ，x_1 の1枝は葉ノードの1を指し，0枝は x_2 のノードを指すようにし，x_2 のノードは1枝は葉ノードの1を指し，0枝は x_3 のノードを指すように x_l まで作れば，F を表す順序付き二分木 T_F ができる．h,i はともに1以上の整数とする．$m_1,m_2,\cdots,m_h$ でラベル付けされた変数ノードは f_q において x_{k_a} または x_{k_b} と項を成す変数ノード，$n_1,n_2,\cdots,n_i$ でラベル付けされた変数ノードは f_q において x_{k_a} または x_{k_b} と項を成さない変数ノードである．Streetに現れる変数は $m_1,m_2,\cdots,m_h$ のいずれか，そして最後に x_{k_a}，または x_{k_b} が現れる． □

(→) F がordered tree-shellableならば，$x_1,x_2,\cdots,x_l$ のノードは street に現れないため，$K_1,K_2,\cdots,K_l$ に対応する1パスはまず x_{k_a},x_{k_b} の1枝を通り，その先の部分木は，論理式 $x_1+x_2+x_3+\cdots+x_l$ を表している．ここには $K_1,K_2,\cdots,K_l$ に対応する1パス以外は存在せず，他の1パスが分岐していることはありえない．この部分木以外に現れることもない．すると，この部分の1パスを削除し，x_{k_a},x_{k_b} の1枝の先を葉ノードの1を指すようにしてできた木は，$f_q+x_{k_a}x_{k_b}$ が ordered tree-shellable となる証拠となる木である． □

6 まとめと考察

本稿では，quadratic 論理関数に変数の数が定数個の積項が一つ加わった形の論理関数が ordered tree-shellable であるかどうかの判定時間は quadratic 論理関数の場合とほぼ同じ時間で判定が可能である．また，二つ加わった場合でも，一つの場合より計算時間を必要とするものの，多項式時間で判定可能であると示した．今後は quadratic 論理関数に加える積項の数をさらに増やした場合の多項式時間判定アルゴリズムを考えること，さらには，積項の数，一積項に現れる変数の最大値など，どのように制限を緩めていけばNP完全になるのかの解明が必要だろう．

参考文献

[1] Y.Takenaga, K.Nakajima, S.Yajima, "Tree-Shellability of Boolean Functions" Theoretical Computer Science,vol.262,pp.633-647.(2001).

[2] C.Benzaken,Y.Crama,P.Duchet,P.L.Hammer and F.Maffray, "More Characterizations of Triangulated Graphs" Journal of Graph Theory 14,pp.413-422.(1990).

[3] Martin Charles Golumbic, "Algorithmic Graph Theory and Perferct Graphs" ACADEMIC PRESS.(1980)

[4] 中田景子,"Tree-Shellable 論理関数の判定の複雑さ", 電気通信大学卒業論文 (2002).

[5] 門野伸史,"Tree-Shellable 論理関数の判定の複雑さ", 電気通信大学修士論文 (2001).